Módulo 6. Javascript & POO

Lección 2 - Ejemplos y ejercicios

Tabla de contenido

Lección 2 - Arreglos

Un array es una lista ordenada de valores a los que te refieres con un nombre y un índice.

Por ejemplo, considera un arreglo llamado emp, que contiene los nombres de los empleados indexados por su id de empleado numérico. De tal modo que emp[0] sería el empleado número cero, emp[1] el empleado número uno, y así sucesivamente.

JavaScript no tiene un tipo de dato array explícito. Sin embargo, puedes utilizar el objeto Array predefinido y sus métodos para trabajar con arreglos en tus aplicaciones. El objeto Array tiene métodos para manipular arreglos de varias formas, tal como unirlos, invertirlos y ordenarlos. Tiene una propiedad para determinar la longitud del arreglo y otras propiedades para usar con expresiones regulares.

Crear un arreglo

Las siguientes declaraciones crean arreglos equivalentes:

```
let arr = new Array(element0, element1, ..., elementN)

let arr = Array(element0, element1, ..., elementN)

let arr = [element0, element1, ..., elementN]
```

element0, element1, ..., elementN es una lista de valores para los elementos del arreglo. Cuando se especifican estos valores, el arreglo se inicia con ellos como elementos del arreglo. La propiedad length del arreglo se establece en el número de argumentos.

La sintaxis de corchetes se denomina "arreglo literal" o "iniciador de arreglo". Es más corto que otras formas de creación de arreglos, por lo que generalmente se prefiere. Consulta Arreglos literales para obtener más detalles.

Para crear un arreglo con una longitud distinta de cero, pero sin ningún elemento, se puede utilizar cualquiera de las siguientes:

```
// Esta...
let arr = new Array(arrayLength)

// ...da como resultado el mismo arreglo que este
let arr = Array(arrayLength)



// Esto tiene exactamente el mismo efecto
let arr = []
arr.length = arrayLength
```

Además de una variable recién definida como se muestra arriba, los arreglos también se pueden asignar como una propiedad a un objeto nuevo o existente:

```
let obj = {}
// ...
obj.prop = [element0, element1, ..., elementN]

// O
let obj = {prop: [element0, element1, ...., elementN]}
```

Si deseas iniciar un arreglo con un solo elemento, y el elemento resulta ser un Número, debes usar la sintaxis de corchetes. Cuando se pasa un solo valor Number al constructor o función Array(), se interpreta como un arrayLength, no como un solo elemento.

```
let arr = [42]      // Crea un arreglo con un solo elemento:
                    // el número 42.

let arr = Array(42)  // Crea un arreglo sin elementos
                    // y arr.length establecidos en 42.
```

```
                //
                // Esto es equivalente a:
let arr = []
arr.length = 42
```

Refiriéndose a elementos del arreglo

Dado que los elementos también son propiedades, puedes acceder a ellos usando la propiedad accessors. Supongamos que defines el siguiente arreglo:

```
        let myArray = ['Wind', 'Rain', 'Fire']
```

Puedes referirte al primer elemento del arreglo como myArray[0], al segundo elemento del arreglo como myArray[1], etc... El índice de los elementos comienza en cero.

Llenar un arreglo

Puedes llenar un arreglo asignando valores a sus elementos. Por ejemplo:

```
let emp = []
emp[0] = 'Casey Jones'
emp[1] = 'Phil Lesh'
emp[2] = 'August West'
```

Acceso a los elementos de una matriz

```
const cars = ["Saab", "Volvo", "BMW"];
let car = cars[0];
```

Cambio de un elemento de la matriz

Cambia el primer elemento de la matriz

```
cars[0] = "Opel";
```

Propiedad longitud de una matriz

Devuelve el número de elementos de una matriz

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let length = fruits.length;
```

Elementos de una matriz en bucle

Una operación común es iterar sobre los valores de un arreglo, procesando cada uno de alguna manera. La forma más sencilla de hacerlo es la siguiente:

```
let colors = ['red', 'green', 'blue']
for (let i = 0; i < colors.length; i++) {
  console.log(colors[i])
}
```

Si sabes que ninguno de los elementos de tu arreglo se evalúa como false en un contexto booleano, si tu arreglo consta solo de nodos DOM (en-US), por ejemplo, puedes usar un lenguaje eficiente:

```
let divs = document.getElementsByTagName('div')
for (let i = 0, div; div = divs[i]; i++) {
  /* Procesar div de alguna manera */
}
```

Esto evita la sobrecarga de verificar la longitud del arreglo y garantiza que la variable div se reasigne al elemento actual cada vez que se realiza el bucle para mayor comodidad.

El método forEach() proporciona otra forma de iterar sobre un arreglo:

```
let colors = ['red', 'green', 'blue']
colors.forEach(function(color) {
  console.log(color)
})
// red
// green
// blue
```

Alternativamente, puedes acortar el código para el parámetro forEach con las funciones de flecha ES2015:

```
let colors = ['red', 'green', 'blue']

colors.forEach(color => console.log(color))

// red

// green

// blue
```

La función pasada a forEach se ejecuta una vez por cada elemento del arreglo, con el elemento de arreglo pasado como argumento de la función. Los valores no asignados no se iteran en un bucle forEach.

Ten en cuenta que los elementos de un arreglo que se omiten cuando se define el arreglo no se enumeran cuando lo itera forEach, pero se enumeran cuando undefined se ha asignado manualmente al elemento:

```
let array = ['first', 'second', , 'fourth']


array.forEach(function(element) {
  console.log(element)
})
// first
// second
// fourth


if (array[2] === undefined) {
  console.log('array[2] is undefined')  // true
}


array = ['first', 'second', undefined, 'fourth']


array.forEach(function(element) {
  console.log(element)
```

```
})
// first
// second
// undefined
// fourth
```

Dado que los elementos de JavaScript se guardan como propiedades de objeto estándar, no es recomendable iterar a través de arreglos de JavaScript usando bucles for...in, porque se enumerarán los elementos normales y todas las propiedades enumerables.

Métodos de array

Método push:

Agregar elementos a una matriz

```
const fruits = ["Banana", "Orange", "Apple"];
fruits.push("Lemon");  // Adds a new element (Lemon) to fruits
```

concat()

Une dos o más arreglos y devuelve un nuevo arreglo.

```
let myArray = new Array('1', '2', '3')
myArray = myArray.concat('a', 'b', 'c')
// myArray is now ["1", "2", "3", "a", "b", "c"]
```

join(delimiter = ',') une todos los elementos de un arreglo en una cadena.

```
let myArray = new Array('Viento', 'Lluvia', 'Fuego')
let list = myArray.join('-')   // la lista es "Viento - Lluvia - Fuego"
```

sort()

Ordena los elementos de un arreglo en su lugar y devuelve una referencia al arreglo.

```
let myArray = new Array('Viento', 'Lluvia', 'Fuego')
myArray.sort()
// ordena el arreglo para que myArray = ["Fuego", "Lluvia", "Viento"]
```

sort() también puede tomar una función retrollamada para determinar cómo se comparan los elementos del arreglo.

El método sort (y otros a continuación) que reciben una retrollamada se conocen como métodos iterativos, porque iteran sobre todo el arreglo de alguna manera. Cada uno toma un segundo argumento opcional llamado thisObject. Si se proporciona, thisObject se convierte en el valor de la palabra clave this dentro del cuerpo de la función retrollamada. Si no se proporciona, como en otros casos en los que se invoca una función fuera de un contexto de objeto explícito, this se referirá al objeto global (window) cuando se usa la función de flecha como retrollamada, o undefined cuando se usa una función normal como retrollamada.

La función retrollamada se invoca con dos argumentos, que son elementos del arreglo.

La siguiente función compara dos valores y devuelve uno de tres valores:

Por ejemplo, lo siguiente se ordenará por la última letra de una cadena:

```
let sortFn = function(a, b) {
  if (a[a.length - 1] < b[b.length - 1]) return -1;
  if (a[a.length - 1] > b[b.length - 1]) return 1;
  if (a[a.length - 1] == b[b.length - 1]) return 0;
}
myArray.sort(sortFn)
// ordena el arreglo para que myArray = ["Viento", "Fuego", "Lluvia"]
```

Si a es menor que b por el sistema de clasificación, devuelve -1 (o cualquier número negativo)

Si a es mayor que b por el sistema de clasificación, devuelve 1 (o cualquier número positivo)

Si a y b se consideran equivalentes, devuelve 0.

Arreglos multidimensionales

Los arreglos se pueden anidar, lo cual significa que un arreglo puede contener otro arreglo como elemento. Usando esta característica de los arreglos de JavaScript, se pueden crear arreglos multidimensionales.

El siguiente código crea un arreglo bidimensional.

```
let a = new Array(4)
for (let i = 0; i < 4; i++) {
 a[i] = new Array(4)
 for (let j = 0; j < 4; j++) {
  a[i][j] = '[' + i + ', ' + j + ']'
 }
}
```

Este ejemplo crea un arreglo con las siguientes filas:

```
Row 0: [0, 0] [0, 1] [0, 2] [0, 3]
Row 1: [1, 0] [1, 1] [1, 2] [1, 3]
Row 2: [2, 0] [2, 1] [2, 2] [2, 3]
Row 3: [3, 0] [3, 1] [3, 2] [3, 3]
```

Usar arreglos para almacenar otras propiedades

Los arreglos también se pueden utilizar como objetos para almacenar información relacionada.

```
const arr = [1, 2, 3];
arr.property = "value";
console.log(arr.property);  // Registra "value"
```

Ejercicio

En el siguiente documento html se indican en comentarios los pasos que debes completar.

```
<!DOCTYPE html>

<html lang="en">

<head>

        <meta charset="UTF-8">

        <title>08 - arrays</title>

        <meta name="author" content="francesc ricart"/>

</head>

<body>


<script>

        var amigos  = ["pedro","maria","joan","pili"];


        // 1- queremos que escriba "mis amigos son maria y joan. rellena los huecos."

        document.write("mis amigos son"+ "<b>" + amigos[] + "</b>" + "  y <b>" + amigos[]+"</b>");


        // 2- ¿dónde está el fallo?

        var enemigos = ["scipion" + "scorpio" + "black"];


        // 3- escribe en el documento web "scorpio es peor que black" de modo que scorpio esté en negrita y la letra sea de color rojo.


        // 4- escribe una instrucción que devuelva la longitud del array amigos sea cual sea el número de datos almacenados en su interior.


        // 5- "aitana" es también tu amiga. Escribe una instrucción que la añada al final de la lista amigos
```

// 6 - "pedro" ha cambiado de nombre. ahora se llama "nacho". actualiza con una instrucción tu lista de amigos

// 7 - "tor" es ahora tu enemigo. añádelo con una instrucción al final de tu lista de enemigos.

</script>

</body>

</html>